(12) UK Patent Application (19) GB (11) 2 338 870 (13) A

(43) Date of A Publication 29.12.1999

(21) Application No 9910222.0

(22) Date of Filing 05.05.1999

(30) Priority Data
(31) 6979898 (32) 01.06.1998 (33) AU
(31) 09139618 (32) 21.08.1998 (33) US

(71) Applicant(s)
Yong-Cong Chen
3/46 Belsize Ave, Carnegie, Vic 3163, Australia

(72) Inventor(s)
Yong-Cong Chen

(74) Agent and/or Address for Service
Charlie Wang
5 Delderfield, LEATHERHEAD, Surrey, KT22 8UA, United Kingdom

(51) INT CL[6]
H04L 12/16 , G06F 17/30 , H04L 29/06

(52) UK CL (Edition Q)
H4P PPG

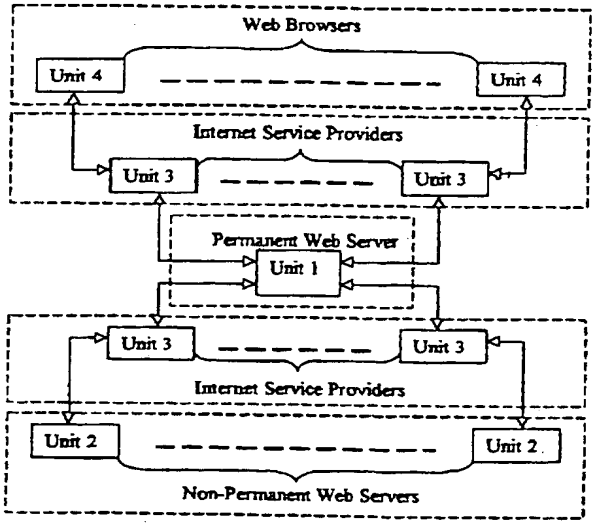
(56) Documents Cited
GB 2333670 A GB 2333427 A WO 98/45781 A1

(58) Field of Search
UK CL (Edition Q) H4P PPG PPS PX
INT CL[6] G06F 17/30 , H04L 12/12 12/16 29/06 29/12
ONLINE : EPODOC, WPI, JAPIO

(54) Abstract Title
Network of distributed, non-permanent, and human interactive web servers

(57) A systematic method of networking and indexing a distributed Web system of non-permanent, human interactive Web servers is described. Typically, the non-permanent Web servers 2 will be hosted on PCs running Windows 9x or NT with dial-up connections to the Internet. The method comprises the following steps. (a) Maintain a permanent central Web server 1 that serves as the network entry point for Web browsers 4 on the Internet. (b) Each non-permanent member server reports its current IP address, search indexes, and other information to the central server every time the member turns active. (c) The central server provides a search service for its active members, which redirects, based on the search results, the Web browsers to the relevant servers using their current IP addresses. (d) Redirection occurs automatically when a browser requests a sub-directory / file whose name matches an active server. (e) Provide for the non-permanent servers a means of maintaining the validity of their entries on search engines elsewhere. (f) Provide a means of using the standard HTML and CGI protocols to conduct and customize instant message-based and / or vocal contacts between site operators and visitors.

Figure 1. Network of Non-permanent Web Servers



GB 2 338 870 A

2338870

Figure 1. Network of Non-permanent Web Servers

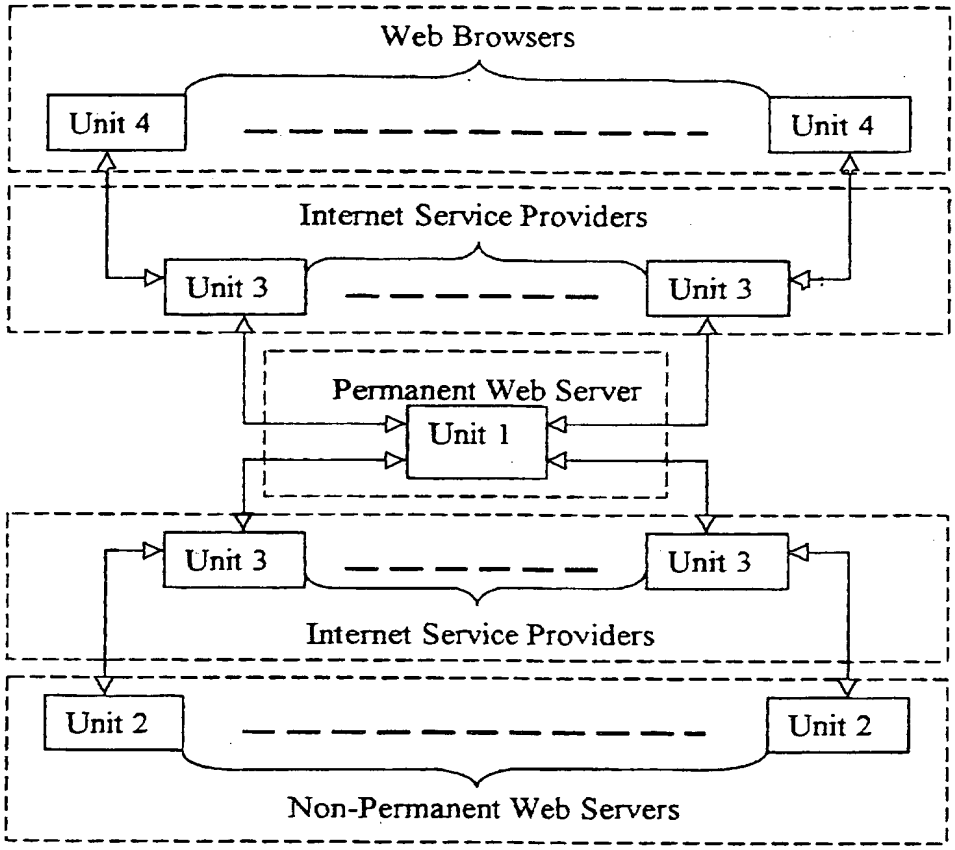


Figure 2. Human Interactive Process

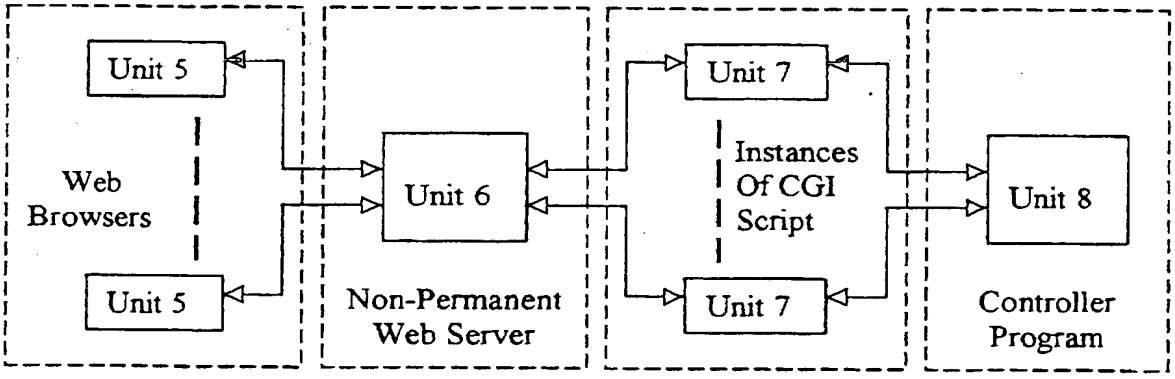


Figure 3. Central Server Database

The following tables, along with a registered system database source name (DNS) "CyberOffice" are assumed in the sample programs, Codes 1 to 3.

Table 1: Search Index

Field Name	Type	Description
Server_Name	Text	A name identifying the server.
Keyword	Text	A keyword describing a directory entry.
Directory	Text	A target directory on the member server.
Server_ID	Number	A unique number identifying the member Web server used across all tables.

Primary Key: Keyword, Server_Name, Directory; **Sort Order**: Ascending

Table 2: User Info

Field Name	Type	Description
Server_ID	Number	A unique number identifying the member Web server used across all tables.
Server_Name	Text	A unique name identifying the serve.
IP_Address	Text	An IP address appended with ":" + the server's TCP port to identify the full location of the server.
User_Name	Text	A name identifying the user of the server.
User_ID	Text	Additional information identifying the user or the user's computer; e.g. license number.
Password	Text	A password for login.
Update_Time	Date/Time	A date-time when the IP_Address was last updated.
Display_Text	Memo	A default text describing the server.
Login_Message	Memo	A Message sent to the user upon login.

Primary Key: Server_ID; **Sort Order**: Ascending

Code 1. User Login Program

```
<% @Language=VBScript %>
<%    'This page is used for both collecting and
      'processing the user login information %>
<% Option Explicit %>
<% Dim i, bSubmit, sUserName, sUserID, sPassword %>
<% Dim sDisplayText, sIPAddress, sLoginMessage, sUpdateTime %>
<%
      Response.Expires = 0
      'bSubmit is set to true when processing the form.
      IF Request.ServerVariables("HTTP_METHOD") = "POST" Then
            bSubmit = TRUE
            'Put user information into variables.
            sIPAddress = Request.ServerVariables("REMOTE_ADDR")
            sUserName = Request.Form("UserName")
            sUserID  = Request.Form("UserID")
            sPassword = Request.Form("Password")
            sDisplayText  = Request.Form("DisplayText")
            'Extract the server port from the UserID field
            'It is appended with a delimeter ":"
            i = InStr(sUserID, ":")
            If i > 1 Then
                  sIPAddress = sIPAddress & Mid(sUserID, i)
                  sUserID = Left(sUserID,i-1)
            End If
            Call VerifyUserLogin
            'The result is set in Session("HasLogin").
      Else
            bSubmit = FALSE
            Session("HasLogin") = false
      End If
%>
<%
'This routine verifies and updates the server information.
Sub VerifyUserLogin()
Dim conn,rs,sql
      'Set the database connection via ODBC
      If IsObject(Session("CyberOffice_conn")) Then
            Set conn = Session("CyberOffice_conn")
      Else
            Set conn = Server.CreateObject("ADODB.Connection")
            conn.open "CyberOffice","",""
            Set Session("CyberOffice_conn") = conn
      End If
      'Format the SQL string.
      sql = "SELECT * FROM [User Info] Where [User_Name] = '" + _
            sUserName + "'"
      sql = sql + " And [Password] = '" + sPassword + "'"
      sql = sql + " And [User_ID] = '" + sUserID + "'"

      'Open the query
      Set rs = Server.CreateObject("ADODB.Recordset")
      rs.Open sql, conn, 3, 3
      If Not rs.eof Then
            Session("HasLogin") = true
            If Not IsNull(rs.Fields("Login_Message").Value) then
                  sLoginMessage = rs.Fields("Login_Message").Value
            End If
            rs.Fields("IP_Address").Value = sIPAddress
```

```
            rs.Fields("User_ID").Value = sUserID
            rs.Fields("Display_Text").Value = sDisplayText
            rs.Fields("Update_Time").Value = NOW
            rs.Update
      Else
            Session("HasLogin") = false
      End If
      rs.Close
      conn.Close
End Sub
%>
<HTML>
<HEAD><TITLE> CyberOffice Login </TITLE></HEAD>
<BODY>
<% If Not bSubmit Then
'The page is loaded for submitting information %>
      <FORM Name=Login METHOD=POST>
      <p align = center > <font size="4" face="Arial, Helvetica">
      <b>CyberOffice Login Sheet</b></font> </p>
      <p> Please complete the following information </p>
      <table>
            <tr>   </tr>
            <tr> User Name: </tr>
            <tr><Input ID=UserName Type=Text Name=UserName> </tr>
            <tr> Password:</tr>
            <tr> <Input Type=Password Name=Password> </tr>
            <tr>   </tr>
            <tr> A short description of your site: </tr>
            <tr> <TextArea Name=DisplayText Rows=8 Cols=40
                  WRAP=VIRTUAL>(Enter the text here)</TextArea></tr>
            <tr>   </tr>
            <tr>
            <p align=right>
            <Input ID=UserID Type=Hidden Name=UserID Value="0">
            <INPUT Type=Submit Value="Submit">   
            <INPUT Type=Reset Value="Reset">
            </p>
            </tr>
            </table>
      </FORM>
<%Else 'Display the login result %>
      <p align = center > <font size="4" face="Arial, Helvetica">
      <b>CyberOffice Login Confirmation</b></font> </p>
      <Table>
      <tr><td NOWRAP> User Name:</td>
            <td> <%Response.Write sUserName%> </td>
      </tr>
      <tr><td NOWRAP> Login Time: </td>
            <td><%Response.Write FormatDateTime(Now)%></td>
      </tr>
      <% If Session("HasLogin") Then %>
            <tr><td NOWRAP> Login Status: </td>
                  <td><Input ReadOnly ID=UserLogin
                  Name=UserLogin Value="Success"></td>
            </tr>
            <tr><td>   </td>
                  <td>   </td>
            </tr>
            <tr><td NOWRAP VALIGN=TOP> Login Message:</td>
                  <td> <TextArea Name=SWContent Rows=10 Cols=30
                  WRAP=VIRTUAL><%Response.Write sLoginMessage%>
```

```
                    </TextArea></td>
              </tr>
              </tr>
        <%Else 'Login fails %>
        <%sLoginMessage = "Please enter the correct information."%>
              <tr> <td  NOWRAP> Login Status: </td>
                    <td><Input ReadOnly ID=UserLogin
                    Name=UserLogin Value="Fail"></td>
              </tr>
              <tr> <td>   </td> <td>   </td> </tr>
              <tr> <td NOWRAP VALIGN=TOP> Login Message: </td>
                    <td> <TextArea Name=SWContent Rows=10 Cols=30
                    WRAP=VIRTUAL><%Response.Write sLoginMessage%>
                    </TextArea></td>
              </tr>
        <% End If %>
        </Table>
<% End If %>
</BODY>
</HTML>
```

Code 2. Auto Redirect Program

```
<%@ LANGUAGE="VBSCRIPT" %>
<%'The custom object detects the existence of an Internet file.
  'The current ASP does not provide a built-in function for it. %>
<OBJECT RUNAT=Server ID=cgi PROGID="ASPControl.InetUtility"></OBJECT>
<% Option Explicit %>
<% Const time_out = 10 'minutes %>
<% Const test_target = "dummy.gif" %>
<% Dim PathInfo, i, Error, sURL, sServerURL, sServerName %>
<%Dim conn,rs,sql
      Error = -1 'no error at all
      Response.Expires = 0
      'Setup database connection via OBDC
      If IsObject(Session("CyberOffice_conn")) Then
            Set conn = Session("CyberOffice_conn")
      Else
            Set conn = Server.CreateObject("ADODB.Connection")
            conn.open "CyberOffice","",""
            Set Session("CyberOffice_conn") = conn
      End If

      'Collect, parse, and process the path information.
      PathInfo = Request.ServerVariables("PATH_INFO")
      i = InstrRev(PathInfo, "/", -1, 1)
      If i > 0 Then
            PathInfo = Left(PathInfo,i)
      End If
      'The top-level sub-directory is the server name.
      i = InStr(2,PathInfo, "/",1)
      If i > 0 Then
            sServerName = Left(PathInfo,i-1)
      Else
            sServerName = PathInfo
      End If
      sServerName = Mid(sServerName, 2)

      FindServer 'The result is set in the sServerURL variable
      If Len(sServerURL) = 0 Then
            Error = 0
      Else
            i = InStr(2, PathInfo,"/")
            If i > 0 Then
                  sURL = sServerURL & Mid(PathInfo, i)
            Else
                  sURL = sServerURL & "/"
            End If
            If TestServer(sURL) = False Then
                  Error = 1
            End If
      End If

      'Set the appropriate actions.
      If Error = -1 Then 'which means no errors at all.
            'Change from the default "302 Object Moved".
            Response.Status ="200 OK"
            Response.Redirect sURL
      ElseIf Error = 0 Then 'The server is inactive.
            Response.Status = "200 OK"
      ElseIf Error = 1 Then 'The destination does not exist.
```

```
            Response.Status = "400 Resource not found"
      End If
%>
<%
'This routine finds the server's URL from the server name.
Sub FindServer()
      sServerURL = ""
      sql = "SELECT * FROM [User Info] Where [Server_Name] = '" + _
            sServerName + "'"
      Set rs = Server.CreateObject("ADODB.Recordset")
      rs.Open sql, conn, 3, 3
      If Not rs.eof Then
            'An inactive server's IP_Address = "0.0.0.0"
            If rs.Fields("IP_Address") <> "0.0.0.0" Then
                  sServerURL = "http://" + _
                        rs.Fields("IP_Address").Value
                  If (Now - rs.Fields("Update_Time").Value) * 24*60 _
                        > time_out Then
                        'time out occurs. Test the server.
                        If TestServer(sServerURL + "/") Then
                              rs.Fields("Update_Time").Value = Now
                        Else
                              rs.Fields("IP_Address").Value="0.0.0.0"
                              sServerURL = ""
                        End If
                        rs.Update
                  End If
            End If
      End If
      rs.Close
End Sub
%>
<%
'This function verifies the existence of a URL.
Function TestServer(sTestURL)
      If IsObject(cgi) Then
            If cgi.VerifyURL(sTestURL & test_target) then
                  TestServer = true
            Else
                  TestServer = false
            End If
      End If
End Function
%>
<HTML>
<HEAD> <TITLE>CyberOffice Auto Response</TITLE> </HEAD>
<BODY>
<% If Error = 1 Then %>
      Error: 400 Resource not found in the destination server.
<% ElseIf Error = 0 Then %>
      <p> We are sorry; the resource you have requested is currently
      unavailable. </p>
      <p> The destination server is temporarily out of service. </p>
<% End If %>
</BODY>
</HTML>
```

Code 3. Index Search Program

```
<%@ LANGUAGE="VBSCRIPT" %>
<%'The custom object detects the existence of an Internet file.
  'The current ASP does not provide a built-in function for it. %>
<OBJECT RUNAT=Server ID=cgi PROGID="ASPControl.InetUtility"></OBJECT>
<% Option Explicit %>
<% Const time_out = 10  'minutes %>
<% Const max_item_number = 100 %>
<% Const test_target = "dummy.gif" %>
<% Const item_prefix = "<LI> <A HREF='" %>
<% Const item_separator = "'>"%>
<% Const item_suffix = "</A>" %>

<% Dim sKeyWord, sResult(), lResult %>
<% Dim conn,rs,sql%>
<% Dim sURL, sServerURL, sServerName, sServerID, sDirectory %>
<%
      Response.Expires = 0
      'Pass in the keyword from either a form or a query string:
      sKeyword = ""
      sKeyWord = Request.Form("KeyWord")
      If sKeyword = "" Then
            sKeyWord = Request.QueryString("KeyWord")
      End If
      'Set up database connection via OBDC:
      If IsObject(Session("CyberOffice_conn")) Then
            Set conn = Session("CyberOffice_conn")
      Else
            Set conn = Server.CreateObject("ADODB.Connection")
            conn.open "CyberOffice","",""
            Set Session("CyberOffice_conn") = conn
      End If
      If IsObject(conn) Then IndexSearch
%>
<%
'The main index search subroutine.
Sub IndexSearch()
Dim sPrevID
      'Format the SQL string.
      'If no keyword is specified, list all active servers.
      If sKeyWord <> "" Then
            sql = "SELECT * FROM [Search Index] Where [KeyWord] = '"
_
                  + sKeyWord + "'"
      Else
            sql = "SELECT DISTINCT [Server_Name], [Server_ID] FROM "
            sql = sql + "[Search Index] "
      End If
      sql = sql  + " ORDER BY [Server_Name]"
      Set rs = Server.CreateObject("ADODB.Recordset")
      rs.Open sql, conn, 3, 3
      lResult = 0
      sPrevID = -1 'A valid ID starts from 0.
      Do Until rs.eof
            sServerID = rs.Fields("Server_ID").Value
            If sServerID <> sPrevID Then
                  sPrevID = sServerID
                  sServerName = rs.Fields("Server_Name").Value
                  If sKeyword <> "" Then
```

```
                        sDirectory = rs.Fields("Directory").Value
                        FindServerURL
                        'The Result is returned in sServerURL.
                  End If
            End If
            If sServerURL <> "" Then
                  sURL = "/"+ sServerName
                  If sKeyWord <> "" Then
                        sURL = sURL + sDirectory
                  Else
                        sURL = sURL + "/"
                  End If
                  Redim Preserve sResult(lResult)
                  sResult(lResult) = sURL
                  lResult = lResult + 1
            End If
            'Set a cutoff when there are too many matches.
            If lResult = max_item_number Then
                  Exit Do
            End If
            rs.MoveNext
      Loop
      rs.Close
      End Sub
%>
<%
'This routine verifies and updates the server's IP address.
'An inactive server is marked with IP_Address = "0.0.0.0"
'Note that the port number is included in the IP_Address field.
Sub FindServerURL()
Dim rsIP
      sServerURL = ""
      sql = "SELECT * FROM [User Info] Where [Server_ID] = " + _
            CStr(sServerID)
      Set rsIP = Server.CreateObject("ADODB.Recordset")
      rsIP.Open sql, conn, 3, 3
      If Not rsIP.eof Then
            If rsIP.Fields("IP_Address") <> "0.0.0.0" Then
                  sServerURL = "http://" +
                              rsIP.Fields("IP_Address").Value
                  If (Now - rsIP.Fields("Update_Time").Value)*24*60 _
                        > time_out Then
                        If TestServer(sServerURL + "/") Then
                              rsIP.Fields("Update_Time").Value = Now
                        Else
                              rsIP.Fields("IP_Address").Value = _
                                                      "0.0.0.0"
                              sServerURL = ""
                        End If
                        rsIP.Update
                  End If
            End If
      End If
      rsIP.Close
End Sub
%>
<%
This routine verifies the existence of a URL.
Function TestServer(sTestURL)
      If IsObject(cgi) Then
            If cgi.VerifyURL(sTestURL & test_target) then
```

```
                    TestServer = true
                Else
                    TestServer = false
                End If
        End If
End Function
%>
<HTML>
<HEAD> <TITLE>CyberOffice Search Result </TITLE> </HEAD>
<BODY>
        <p> List of items that match "<%Response.write sKeyword%>":
</p>
<% If lResult = 0 Then %>
        <p> No matchs are found for the keyword. </p>
<%Else %>
<DIR>
<%
Dim i
        For i = 0 To lResult -1
            Response.write item_prefix + sResult(i)+ item_separator _
                            + sResult(i) + item_suffix
        Next
%>
</DIR>
<% End If %>
</BODY>
</HTML>
```

Code 4. ***Visual Basic Class Module for "ASPControl.InetUtility"***

```
Const LINE_BUFFER = 255
Const ASP_CONTROL = "ASPControl"
Const INTERNET_OPEN_TYPE_DIRECT = 1
Const INTERNET_FLAG_EXISTING_CONNECT = &H20000000
Const INTERNET_FLAG_NO_CACHE_WRITE = &H4000000
Const INTERNET_FLAG_DONT_CACHE = &H4000000
Const HTTP_QUERY_STATUS_CODE = 19

#If UNI_CODE = 1 Then 'Unicode Declarations for Internet Functions
Private Declare Function InternetOpen Lib "WinINet.Dll" Alias _
"InternetOpenW" ( _
    ByVal lpszAgent As String, ByVal dwAccessType As Long, _
    ByVal lpszProxy As String, ByVal lpszProxyBypass As String, _
    ByVal dwFlags As Long) As Long
Private Declare Function InternetOpenUrl Lib "WinINet.Dll" Alias _
"InternetOpenUrlW" ( _
    ByVal hInternet As Long, _
    ByVal lpszUrl As String, _
    ByVal lpszHeaders As String, _
    ByVal dwHeadersLength As Long, _
    ByVal dwFlags As Long, _
    ByVal dwContext As Long) As Long

Private Declare Function HttpQueryInfo Lib "WinINet.Dll" Alias _
"HttpQueryInfoW" ( _
    ByVal hRequest As Long, _
    ByVal dwInfoLevel As Long, _
    ByVal lpBuffer As String, _
    ByRef lpdwBufferLength As Long, _
    ByRef lpdwIndex As Long) As Long
#Else 'Regular Declaration used in Windows 9x and NT.
Private Declare Function InternetOpen Lib "WinINet.Dll" Alias _
"InternetOpenA" ( _
    ByVal lpszAgent As String, ByVal dwAccessType As Long, _
    ByVal lpszProxy As String, ByVal lpszProxyBypass As String, _
    ByVal dwFlags As Long) As Long
Private Declare Function InternetOpenUrl Lib "WinINet.Dll" Alias _
"InternetOpenUrlA" ( _
    ByVal hInternet As Long, _
    ByVal lpszUrl As String, _
    ByVal lpszHeaders As String, _
    ByVal dwHeadersLength As Long, _
    ByVal dwFlags As Long, _
    ByVal dwContext As Long) As Long
Private Declare Function HttpQueryInfo Lib "WinINet.Dll" Alias _
"HttpQueryInfoA" ( _
    ByVal hRequest As Long, _
    ByVal dwInfoLevel As Long, _
    ByVal lpBuffer As String, _
    ByRef lpdwBufferLength As Long, _
    ByRef lpdwIndex As Long) As Long
#End If
Private Declare Function InternetCloseHandle Lib "WinINet.Dll" _
(ByVal hInternet As Long) As Long

'The main function of the class
Public Function VerifyURL(URL As String) As Boolean
Dim bRet As Long, dwBuffer As Long, dwIndex As Long
```

```
    Dim iStatus As Long, i As Long, hSession As Long
    Dim hFile As Long, szBuffer As String, sName As String
          'We are only interested in HTTP resources.
          If InStr(1, URL, "HTTP://", vbTextCompare) <> 1 Then
                VerifyURL = True
                Exit Function
          End If
          szBuffer = String(LINE_BUFFER + 1, Chr(0))
          sName = ASP_CONTROL

          hSession = InternetOpen(sName, INTERNET_OPEN_TYPE_DIRECT, _
                "", "", 0)
          If (hSession) Then
                hFile = InternetOpenUrl(hSession, _
                      URL, "", 0, _
                      INTERNET_FLAG_EXISTING_CONNECT And _
                      INTERNET_FLAG_DONT_CACHE, 0)
          End If
          If (hFile) Then
                dwBuffer = LINE_BUFFER
                bRet = HttpQueryInfo(hFile, _
                      HTTP_QUERY_STATUS_CODE, _
                      szBuffer, _
                      dwBuffer, _
                      dwIndex)
                If (bRet) Then
                      i = InStr(szBuffer, Chr(0))
                      If i > 1 Then szBuffer = Left(szBuffer, i - 1)
                      szBuffer = Trim(szBuffer)
                      i = InStr(szBuffer, " ")
                      If i > 0 Then szBuffer = Left(szBuffer, i - 1)
                      If IsNumeric(szBuffer) Then
                            iStatus = CInt((szBuffer))
                      End If
                      If iStatus = 0 Or iStatus >= 300 Then
                            bRet = False
                      End If
                End If
                Call InternetCloseHandle(hFile)
          End If
          If (hSession) Then Call InternetCloseHandle(hSession)
          If bRet Then
              VerifyURL = True
          End If
    End Function
```

A NETWORK OF DISTRIBUTED, NON-PERMANENT, AND HUMAN INTERACTIVE WEB SERVERS

Field of the Invention

The present invention relates to the general field of client-server communications over the Internet. More specifically, it contains server-side technologies for facilitating a network of distributed, non-permanent, human-interactive Web servers over the Internet.

Background of the Invention

The increased popularity of the worldwide Web (WWW or the Web) and the companion Web browser has revolutionized the global inter-connected network (the Internet) in the past few years. Today there are millions of Web sites and hundreds of millions browsers around the world, which constitutes the phenomenal "Web life". The Internet provides contents-rich information covering almost every aspect of the human world. The Maturing Web technology now offers, to many industries, ideal opportunities for Internet commerce such as on-line trading and on-line services.

Growing activities on the Internet exert a great deal of pressure on the capacity and complexity of Web servers. Typically, a commercial Web server requires at least a fixed Internet Protocol (IP) address and one or more domain names. The server would normally reside on a high-speed computer with sufficient network bandwidth for large numbers of concurrent connections. It would be further backed by some high-end database application. This makes it rather difficult for small business to adopt the full Web life within relatively low budgets.

Challenged by the above difficulty, the disclosed invention introduces the concept of distributed Web system in which the traditional centralized single server is replaced by a cluster of low-end, non-permanent Web servers that would typically reside on desktop PCs with dial-up network connections. This means that everyone is capable of hosting a Web site on his / her everyday PC. Inspired by this perspective, a further embodiment of the invention provides a means of direct message-based and / or vocal contacts between a site operator and Web visitors. The present invention also solves a

number of practical problems with this type of systems, in particular, the search index maintenance for non-permanent servers.

The potential of the disclosed invention would enormously benefit small business and / or businesspersons. Imagine that a home business serves its customers worldwide by simply being on the net, or that a stockbroker talks with his/her clients while displaying their portfolio on-line, not to mention numerous kinds of on-line shops. The future of the Web commerce would be dominated by a large number of specialized Web networks, with each of them presumably having their own focus.

Summary of the Invention

A systematic method of networking and indexing a distributed Web system of non-permanent, human interactive Web servers (sites) is invented. A non-permanent server typically runs on, but not restricted to, a desktop PC with dial-up connection to an Internet service provider. The latter in turn connects to the Internet and assigns, typically, a temporary IP address to the PC. The non-permanent servers are further empowered with on-line human contact facilities, which constitutes an integral part of the invention as described in step (f) below. In accordance with the disclosed invention, the method of networking and managing the distributed system comprises the following steps. (a) Maintain a permanent central Web server that serves as the entry point for Web browsers on the Internet. The central server holds all relevant information about its non-permanent members, which typically includes, but not limited to names, current IP addresses, search indexes, and server directories. (b) Each non-permanent Web server reports to the central server, via e.g. login, its IP address every time it turns active. The login may upload the search indexes or any other associated information. (c) The central server provides Web visitors a search service for active members on the network. Upon selecting an appropriate item, a browser is redirected to the relevant server using its current IP address. (d) The central server interprets and automatically redirects a request to a member server if the request asks for a directory / file whose top-level sub-directory identifies the name of an active member and the subsequent part identifies a directory /file on the server. The program includes a time-out mechanism for periodically verifying / updating the IP addresses of its members. An explanatory / courtesy message is returned when the request is intended for an in-active server. (e) The central server provides a means of

maintaining index entries on search engines elsewhere when a member server is out of service. Normally, if a public search engine fails to locate an Internet resource, it will remove the entry from the engine's database. In its simplest form, but not limited to this, the last part of step (d) offers a natural solution for the problem. In this case the explanatory text may be extended, based on the up-to-date information at the central server, to include dynamically constructed and properly formatted indexing information for that server. (f) Within the standard HyperText Markup Language (HTML) and Common Gateway Interface (CGI) protocols, operators behind the non-permanent servers can conduct direct, instant message-based and / or vocal contacts with visitors. The associated technique is capable of handing several concurrent visitors and transferring / downloading arbitrary files / programs on demands.

Brief Description of the Drawings and Program Codes

The features of the invention will become more readily apparent and may be better understood by referring to the following detailed description and exemplary embodiment of the present invention, taken in conjunction with the accompanying drawings, tables and program codes, in which:

Figure 1 Illustrates, in block diagram format, the disclosed network of Web servers.

Figure 2 Illustrates, in block diagram format, the human interactive process on a non-permanent Web server.

Figure 3 Describes, in table format, a sample Central Server database.

Code 1 Lists a sample User Login program written in Microsoft Active Server Page.

Code 2 Lists a sample Auto Redirect program written in Microsoft Active Server Page.

Code 3 Lists a sample Index Search program written in Microsoft Active Server Page.

Code 4 Lists a Visual Basic class module used in Codes 2 and 3.

Detailed Description

The Web system

Definition: A Web server, or a Web site, in the context of the present invention, always refers to an Internet application that runs on a computer, listens to a pre-defined Transport Control Protocol (TCP) port (preferably port number 80). It must interact with a Web browser application via the standard HyperText Transfer Protocol (HTTP) version 1.1 or above.

Refer to Figure 1, where unit 1 is a permanent Web server on the Internet with a registered domain name, unit 2's are non-permanent servers, unit 3's are Internet service providers (ISP), and unit 4's are Web browsers. Both units 2 and 4 reside on computers that typically have, but not limited to, dial-up connections to the Internet via unit 3. The connections can also be made, for example, through cable modems that do not require dial-up. Unit 1 shall be named the Central Server of the network. It must support all the functions of a regular Web server plus at least one kind of database applications. Unit 2 shall be named the Member Server of the network. It must support the standard CGI specifications version 1.1 or above. The data transmission rate between unit 2 and unit 3 should be greater than the average rate between unit 3 and unit 4. When supporting unit 2, unit 3 must allow the former to run server applications that listen to one or more TCP ports and it should support non-interruptive Internet sessions up to a few hours at any time. There are no special requirements on units 4. The network is "distributed" in the sense that there are no requirements on the servers' physical locations.

The most prominent problem for the network lies in that, a member server, by characteristic, does not have a fixed IP address. Normally, every time a computer starts an Internet session, it is assigned a temporary IP address by the ISP. Once the IP address is obtained, the Web server application may be started. Each time a member server turns on or is back from an interrupted service, it must report to the central server. This may be done by, but not limited to, filling and submitting a special HTML form to the central server. The Web page should contains information such as name, password, user ID, site contents / indexes, and other relevant materials. Upon receiving this information, the central server will save the IP address of the sender,

along with the current time to a database. The time stamp will be used in a time-out mechanism for verifying the IP address to be described below.

The central Web server functions as the gateway of the network to the entire Internet. It servers as an entry point for Web browsers and receives all the initial requests from them. There are in general two relevant types of browser requests submitted by the HTTP "GET" command:

Type A A request that does not specify a file name or a directory other than the default file / directory of the central server.

Type B A request that specifies, with or without a file name, a directory whose top-level sub-directory matches an existing active / inactive member server.

The central server provides a search service for Type A requests. This should be the default page of the server. The visitors may be asked to enter search criteria such as keywords or may select from a list or lists of complete active servers. The search program may use the pre-stored and / or the updated information from each server to carry out the query over the database. The search program should verify each member's IP address in the result list before displaying it. This may be accomplished by a time-out mechanism as follows:

1. A pre-determined time-out value is set. A reasonable value would be 10 minutes.
2. If a given server's IP address has not be updated within the time-out value, the search program will attempt to load a pre-set test target at the server's root directory.
3. If the test target is found, the time-stamp for the IP address will be updated.
4. If the test target is not found, the server will be marked as out of service.
5. Only active servers will be displayed in the search result.

For a Type B request, the central server redirects or replies the request based on the following procedures:

1. Determine the current status of the server to which the request is intended, using the time-out mechanism described above.

2. Optionally verify the existence of the resource on the target server.
3. If the sever is active, redirect the request to the intended destination.
4. Optionally return an error message to the browser if the resource does not exist. This will help search engines on the Internet to update their databases.
5. If the server is out of service, a valid explanatory / courtesy page is returned. The page may include dynamically constructed and properly formatted indexing information suitable for general search engines on the Internet.

The last step may solve a traditional problem for non-permanent Web servers. Namely if a public search engine fails to locate an Internet resource after a certain number of times, it will remove the entry from the database. Note that the address stored in the search engine's database should always point to the permanent central server.

Human Interactive

Definitions: "Human interactive" or "human contact", in the context of the present invention, refers to direct, instant on-line information exchange between a Web operator behind a member server and Web visitors (i.e., people who use Web browsers) through the standard browser-server channel. It should be distinguished from other methods of Internet communications such as email, "write or talk" in some UNIX systems, Internet chat, Internet meeting, and etc. These methods require permanent domain names / addresses or assistance from third party servers. Microsoft in this context always refers to Microsoft Corporation, Redmond, Washington, USA.

Most of the member servers in the Web network are expected to reside on desktop PCs running Microsoft Windows 95, 98, and NT, although they are not limited to the Microsoft platforms. Consequently, the Web operator who runs a member server is likely to stay physically nearby the PC. The Web network therefore is capable of conducting live interactions with visitors. A means of on-line human contacts within the standard HTML and CGI specifications is invented, which constitutes an integral part of the Web network.

Refer to Figure 2, where unit 5's are Web browsers running on top of any platforms, unit 6 is a non-permanent Web server, unit 8 is a server-side controller application, and unit 7's are instances of a CGI script that passes data between the server and the controller application. The interactive processes are managed by the controller

application, which may run on the server's computer or on another computer that links to the server's computer via a local area network (LAN). The controller application should have a well-presented user interface. The process of information exchange undergoes the following steps.

1. An interactive session starts with the browser posting information via a standard HTML form to the Web server.

2. The server invokes and passes the information to the target CGI script specified by the HTML form.

3. The CGI script in turn passes the information to the controller application and waits for response. The communication between the two processes should utilize advanced features of the operating system such as anonymous or named pipe. The two processes should be allowed to run on different computers if there is a LAN.

4. The controller program then presents the information to the operator. The program should be capable of holding concurrent connections from several visitors (with respect to the browsers' IP addresses).

5. The operator may enter response messages to the visitor. The controller program should allow inserting pre-written macros and files. The file insertion may use either HTML <A> or HTML <embed> tags, depending on the nature of the file. The operator may embed live-recorded sound waves and / or other multimedia sources.

6. The response is then formatted into a valid HTML page and is passed back to the CGI script. The page may be another HTML form suitable for continuing the conversation.

7. The CGI script returns the page to the browser via the Web server. The browser may download any additional files specified in the page.

8. Optionally, the operator may invite the visitor to use any of the state-of-art Internet communication tools that are available on both sides. Normally, this can be achieved when they are both running same kinds of operating systems such as Windows 95. In the latter case, the visitor can most conveniently

download an ActiveX control that is readily talking to the server's computer, thus avoiding any third-party servers.

Exemplary Embodiment

In what follows, a full exemplary embodiment of the invention will be presented, which applies to Microsoft Windows 9x and / or NT operating systems. While this exemplary embodiment gives a practical implementation of the disclosed Web system, it is understood by those skilled in Web technology that various modifications in form and detail may be made therein without departing from the scope and spirit of the invention. Accordingly, modifications such as those suggested above and below, but not limited thereto, are to be considered within the scope of the invention.

System Requirements

The above said central Web server may be implemented by Microsoft Internet Information Server (IIS) version 4.0 (or above) running on a Windows NT Server version 4.0 (or higher) operating system. The computer that hosts the server should have at least 64M Random Access Memory (RAM), a Pentium 200 MHz Central Process Unit (CPU) or equivalent CPU of any other brands, and a reliable permanent Internet connection with minimum 64kbps of data transmission rate. The search program may be written in terms of Microsoft Active Web Page (ASP). The latter is a server-side scripting language supported by IIS. The underlying database may be either Microsoft Access 97 (or above) or Microsoft SQL Server 6.5 (or above). The connection between ASP and the database is made through Open Database Connectivity (ODBC). The database must be registered under the 32bit ODBC registry with a system DSN (database source name). The central server can be a virtual Web site provided by an ISP, namely its IP address can be shared by other domain names not relevant to the present invention.

A member server of the network may be implemented using Microsoft Personal Web Server (PWS) version 4.0 or any HTTP server applications that run under Microsoft Windows 95, 98, NT 4.0, NT Server 4.0, or higher versions of Microsoft Windows. The computer that hosts the server should have at least 32M Random Access Memory (RAM), a Pentium 166 MHz CPU or equivalent CPU of any other brands, and a reliable dial-up Internet connection with minimum 33.6kbps of modem speed. It

should optionally have a microphone, speakers, and a sound card. The computer should already have been installed with Microsoft Internet Explorer version 4.01 and Microsoft Windows Socket Implementation version 2.0 or above, both of which are required by PWS. The advantage of using PWS is that it also supports ASP, a convenient tool for creating instant database application. Note that PWS normally starts as one turns on the PC.

On the Center Web Server

A simplest central Web server may use one database and three core programs. The database shall be named "Central Server" and it may contain two tables, a "User Info" table that holds the login information and a "Search Index" table that contains directories and key words on the member servers. The structure of the database is illustrated in Figure 3. Other tables containing account information etc. may be included in the database. The central server maintains most of the information manually in this embodiment. But many of the items can be updated on-line in a more sophisticated system. The programs may be written in ASP pages that interact with the database.

The first program shall be named "User Login". The login compromises the following steps

1. A controller program (cf. below for more discussions) on a member server's computer invokes an HTML form via a standard Web browser.
2. Upon filling in user name, password, user ID and server's TCP port the controller program posts the information back to the User Login program. Extensions can include additional information such as site description, search index, and etc.
3. Upon successful verification of the user identity, the User Login program updates the "IP_address" and "Update_Time" fields in the User Info table. The program extracts the IP address of the member server from the variable Request.ServerVariables ("REMOTE_ADDR") provided by ASP.

A working sample ASP that illustrates both the HTML form and the User Login program is given in Code 1.

The second program shall be named "Auto Redirect". For simplicity, given a member server, only a limited number of directories / sub-directories will be indexed (thus searchable), and they must all include a test target and a default page. Each of the directories is replicated on the central server within a directory whose name identifies the member server. Each replica contains one default file that includes the Auto Redirect program. When a directory is requested, the program executes the following steps:

1. Extract the destined server and obtain its IP address and current status from the database.
2. If the server is out of service, return a default message explaining the situation and exit the program.
3. If a time-out occurs for the IP address, attempt to load the test target under the root directory of the destined server.
4. If the target cannot be loaded, mark the server as inactive then go to step 2. Otherwise update the time-stamp of the IP address.
5. Attempt to load the test target under the intended directory of the destined server.
6. If the target cannot be loaded, return an error message to the browser (because the directory probably does not exist on the destined server) and exit the program.
7. Redirect the browser to the new directory.

The Auto Redirect program does not apply when the request specifies a file. A better-controlled CGI program may be used for the latter case. But the algorithm remains the same. Furthermore, step 2 may be extended to reply informative contents suitable for Internet search engines to obtain indexes. A working sample ASP that illustrates the above steps is given in Code 2.

The third program shall be named "Index Search", which should be called by the default page of the central Web server. The central part of the program is a search routine that compromises the steps:

1. Identify the entries in the Search Index table that match the keyword. Filter out servers that are out of service.

2. If a time-out occurs for an active IP address, attempt to load the test target in the root directory of the destined server.

3. If the test target cannot be loaded, mark the server as inactive. Otherwise update the time-stamp of the IP address.

4. List the output as links to the local directories on the central server (cf. above) so that the browser can store them. The Auto Redirect program will then redirect the browser to the final location.

A working sample ASP that illustrates the above steps is given in Code 3. Code 4 contains the custom class module (object) that verifies an Internet file. The object is used in both Codes 2 and 3.

The above description covers the core ingredients and the programming style of this exemplary embodiment. Within the general guidelines sketched here, a full central Web server may be customized to suit any special requirements for the network.

On the Member Server

A member server's computer must have one root directory for the Web server, which may contains any number of subdirectories. In addition, there may be virtual directories attached to the root directory. Each directory indexed by the central server must contain the test target used to verify the directory. In its simplest form, the member server may use one or more standalone controller program(s) written in C++ or Visual Basic or any other Windows programming languages in conjunction with the Web server application. The controller program may login to the central server at the startup, using the User Login program described above.

The controller program is also responsible for managing the human interactive functions. The CGI script that passes information between the Web server and the controller program(s) may be written in C or C++. It should be compact, runs fast, and allows multiple instances. The communication between the script and the controller program may use anonymous pipe for Windows 9x and named pipe for Windows NT. The latter allows the controller program(s) to reside on another computer in an NT network. In response to a visitor's inquiry, the controller program may insert any source of file(s) on the local disk (network) or on the Internet. The existence of the file(s) should be verified before being inserted. A local file should be made available

on the Internet by, for example, copying it to a temporary place under the server's root directory. The format of the file may be sound waves, Word documents, video clips, images, databases, and etc.

When the remote browser runs on any 32bit Microsoft Windows, the visitor may be invited to use more advanced Internet communication programs such as Microsoft NetMeeting. The visitor may also be instructed to download an ActiveX control that is automatically set to talk to the operator. This eliminates the need for any third party servers. Programming tools that implement Internet phone conversation are widely available in C++, Visual Basic and other Windows programming languages. The control should be code-signed and certified as safe-to-use before being placed on the Internet.

CLAIMS

The claims defining the invention are as follows:

1. A systematic method of networking and indexing a distributed Web system of non-permanent Web servers having possibly dynamic IP addresses, the method comprising the steps of:
   A. Maintaining a permanent central Web server that serves as the network entry point for Web visitors;
   B. Storing in the central server each member server's current IP address, contents index, and other relevant information required by the network;
   C. Providing for non-permanent member servers an index service based upon the stored information;
   D. Automatically redirecting requests made by Web browsers to the relevant active non-permanent member servers when the browsers request such information;
   E. Automatically generating informative explanation pages when the requested non-permanent member servers are inactive; and
   F. Providing for web browser operators who access a non-permanent member server a means of direct message or vocal contacts with the system operator of the member server within the standard HTML and CGI specifications.

2 The ~~method of~~ claim 1 wherein the index service provided by the central Web server further comprises the steps of:

A. ~~Maintaining~~ a database that collects the IP addresses and directory indexes of the ~~non-permanent~~ member servers;

B. Constantly updating the IP addresses with a time-out mechanism;

C. Providing key words search over active non-permanent member servers;

D. Listing the search result in terms of links to permanent locations on the central server that may be stored by Web browsers or Internet search engines; and

E. Automatically redirecting Web browsers from the permanent locations to the intended directories on the non-permanent servers.

3 The method of claim 1 wherein the implementation of the direct contact between a Web browser operator and the system operator of a non-permanent Web compromises the steps of:

A. The Web browser posting information / inquiry from a standard HTML form to a CGI script;

B. The CGI script passing the data to a standalone controller program capable of holding multiple concurrent inquiries;

C. The operator of the controller program returning a HTML reply to the CGI script;

D. The reply page containing optionally embedded/linked files, multimedia sources or downloadable programs to customize the implementation; and

E. The CGI script returning the reply via the server to the Web browser.

The Patent Office

15

INVESTOR IN PEOPLE

Application No: GB 9910222.0 **Examiner:** Ken Long
Claims searched: 1 to 3 **Date of search:** 22 October 1999

Patents Act 1977
Search Report under Section 17

Databases searched:

UK Patent Office collections, including GB, EP, WO & US patent specifications, in:

UK Cl (Ed.Q): H4P (PPG, PPS & PX)

Int Cl (Ed.6): H04L (12/12, 12/16, 29/06 & 29/12)
G06F (17/30)

Other: ONLINE : EPODOC, WPI, JAPIO

Documents considered to be relevant:

Category	Identity of document and relevant passage		Relevant to claims
A	GB 2333670 A	L M ERICSSON	None
A	GB 2333427 A	IBM	None
A	WO 98/45781 A1	FLASH (page 3 lines 11 to 18)	None

X	Document indicating lack of novelty or inventive step	A	Document indicating technological background and/or state of the art.
Y	Document indicating lack of inventive step if combined with one or more other documents of same category.	P	Document published on or after the declared priority date but before the filing date of this invention.
&	Member of the same patent family	E	Patent document published on or after, but with priority date earlier than, the filing date of this application.

r:<GB_____2338870A__I_>